Timo Lausmaa

AUTOMAATTINEN JÄRJESTELMÄ METSÄNKÄYTTÖILMOITUSTEN LUONNILLE

AUTOMAATTINEN JÄRJESTELMÄ METSÄNKÄYTTÖILMOITUSTEN LUONNILLE

Timo Lausmaa
Opinnäytetyö
Syksy 2017
Tietotekniikan koulutusohjelma
Oulun ammattikorkeakoulu

TIIVISTELMÄ

Oulun ammattikorkeakoulu
Tietotekniikan koulutusohjelma, ohjelmistokehitys

Tekijä(t): Timo Lausmaa
Opinnäytetyön nimi: Automaattinen järjestelmä metsänkäyttöilmoitusten luonnille
Työn ohjaaja(t): Lauri Pirttiaho, Lasse Haverinen, Jari Oksa
Työn valmistumislukukausi ja -vuosi: 2017 Sivumäärä: 30

Opinnäytetyö tehtiin Piimega Oy:lle keväällä 2016. Piimega Oy on oululainen mm. verkkokauppa- ja toiminnanohjausjärjestelmiä kehittävä ohjelmistoyritys.

Työn aiheena oli ohjelmoida automaattinen tiedonsiirtojärjestelmä luomaan ja lähettämään metsänkäyttöilmoituksia, joita Piimegan metsäteollisuudelle tuotetun ohjelmiston tyypilliset käyttäjät joutuvat työssään useasti tekemään. Metsänkäyttöilmoitusten lähetys mahdollistui Metsäkeskuksen ylläpitämän web-rajapinnan ansiosta.

Työ toteutettiin uudeksi osaksi olemassa olevaan ohjelmistoon. Työ onnistui kokonaan VB.Netin tarjoamilla kirjastoilla hyvän HTTP- ja web-rajapintojen tuen ansiosta.

Asiasanat: metsänkäyttöilmoitus, HTTP, rajapinta

ABSTRACT

Oulu University of Applied Sciences
Degree programme, Bachelor of Engineering

Author(s): Timo Lausmaa
Title of thesis: Automatic creation of forest use declaration
Supervisor(s): Lauri Pirttiaho, Lasse Haverinen, Jari Oksa
Term and year when the thesis was submitted: 2017 Pages: 30

This thesis was done at Piimega Oy during the spring of 2016. Piimega Oy is a software company in Oulu specializing in, but not limited to ERP and e-commerce solutions.

The goal of the thesis was to create an automatic system for the creation and sending of forest use declarations to a web interface maintained by Metsäkeskus (Forest Centre). A typical user of the forest industry specialized ERP software from Piimega is often required to submit forest use declarations in their daily job. This system was created to simplify and ease that process.

The work was implemented in the VB.Net language by using the many tools it offers for interfaces and HTTP protocol. The created system was implemented into an existing software solution.

Keywords: forest use declaration, HTTP, interface

ALKULAUSE

Haluan kiittää Piimega Oy:tä mielenkiintoisen opinnäytetyön tarjoamisesta sekä Oulun ammattikorkeakoulun ohjaajia Lauri Pirttiahoa ja Lasse Haverista neuvoista opinnäytetyön tekemisessä.

Oulu, 27.11.2017

Timo Lausmaa

SISÄLLYS

SANASTO

RESURSSI: Tietokoneella oleva tiedosto. Esimerkiksi .xml-tiedosto, .jpg-kuva tai tekstitiedosto.

PARAMETRI: Ohjelmalle tai sen osalle välitettävä tieto. Ohjelman ulkopuolelta tulevia parametreja kutsutaan tarkemmin argumenteiksi, sisäisesti välitettävät tiedot ovat parametrejä. Tällaisia tietoja voivat olla esimerkiksi tässä työssä Metsäkeskuksen tiedonsiirtopalvelun tunnistuksessa käytettävä käyttäjätunnus ja salasana.

XML: Extensible Markup Language

HTTP: Hypertext Transfer Protocol

1 JOHDANTO

Opinnäytetyö tehtiin keväällä 2016 Piimega Oy:lle osaksi Piimegan olemassa olevaa toiminnanohjausjärjestelmää. Työ tehtiin muiden samassa yrityksessä suorittamieni harjoittelujen jälkeen, ja aiemmin tekemistäni töistä saadusta kokemuksesta oli apua järjestelmän toteutuksessa.

Yrityksenä Piimega Oy on erikoistunut muun muassa verkkokauppa- ja toiminnanohjausjärjestelmiin, joista jälkimmäisiin kuuluu ForestPro-toiminnanohjausjärjestelmä, jonka osaksi metsänkäyttöilmoitusten lähettämiseen toteutettu järjestelmä tuli. ForestPro-järjestelmä on kehitetty metsäteollisuuteen muun muassa puunhankinnan, korjuun ja logistiikan hallinnoimiseen. Järjestelmällä voidaan myös luoda monia asiakirjoja kaikista yllä mainittujen prosessien vaiheista.

Opinnäytetyön aiheena oli luoda rajapinta, joka osaisi luoda jokaiselle halutulle puunhankintakaupalle ja niille sisältyville lohkoille valmiiksi täytetyn metsänkäyttöilmoituksen ja lähettää sen automaattisesti Metsäkeskuksen ylläpitämään rajapintaan käyttäjän puolesta.

Aikaisemmin metsänkäyttöilmoitusten tekeminen järjestelmällä oli onnistunut tulostamalla asiakirja ohjelmasta osittain täytettynä, minkä jälkeen asiakirjan lopullinen toimittaminen metsäkeskukselle jäi käyttäjän velvollisuudeksi esimerkiksi sähköpostilla tai postitse. Metsänkäyttöilmoituksen lakisääteisyyden vuoksi ohjelmaan haluttiin prosessia helpottava automaattinen järjestelmä vähentämään manuaalisen työn määrää ohjelman käyttäjältä.

Projekti valittiin opinnäytetyön aiheeksi osittain sen tarjoaman prosessin kehitysmahdollisuuden vuoksi. Projektin alussa arvioitiin, että tavallisesti yhden metsänkäyttöilmoituksen täyttäminen veisi arviolta noin 20–30 minuuttia. Hieman manuaalista tietojen syöttöä vaativalla, mutta muuten automatisoidulla järjestelmällä käytetyn ajan voisi lyhentää noin 1–5 minuuttiin, minkä ansiosta pitkällä aikavälillä säästetty aika olisi merkittävä.

Oman kehityksen ja ohjelmoinnin oppimisen kannaltakin työ oli mielestäni hyvä aihevalinta, sillä komponenttina kehitettävä ominaisuus tuli osaksi aiemmin toteutettua serveriohjelmistoa, jonka toimintavarmuus ja vakaus siinä jo entuudeltaan toimivien järjestelmien tärkeyden vuoksi oli erittäin tärkeä. Tämän vuoksi vakaan ja hyvin testatun koodin tuottaminen oli tärkeää. Ilmoitusten lähettämiseen sekä muuhun sisäiseen kirjaamiseen käytetty tekniikka, HTTP POST, on nykyaikana muiden HTTP-pyyntöjen kanssa erittäin tärkeä teknologia muun muassa web- ja mobiiliohjelmistoissa.

2 TYÖN TAUSTA

Piimegalla käytössä olevat sekä yleisesti alalla vallitsevat agile- (suom. ketterä) ja SaaS (software as a service) -kehitysmetodit tarkoittavat jatkuvaa kehitystä, jolle ei ole määritelty loppumisajankohtaa. Ohjelman kehityssykliä pyritään pitämään ketteränä usein tapahtuvilla päivityksillä sekä reagoimalla asiakkaiden ja käyttäjien antamaan palautteeseen uusista ominaisuuksista sekä mahdollisista parannuksista olemassa olevista toiminnoista.

Ohjelma, jota osaksi projekti toteutettiin, on suunnatttu yrityskäyttäjille puunhankinnan hallintaan. Monet ohjelman käyttäjät ovat joko suoraan tai välillisesti tekemisissä puiden hakkuun kanssa ostaessaan puuta ja hallitessaan puun logistiikkaketjua metsästä eteenpäin. Puiden hakkuita sääntelevät Suomen metsälait, jotka määrittävät sääntöjä, miten metsää voi hakata ja miten sitä tulee uudistaa. Metsakeskus.fi -sivulla kerrotaan velvollisuuksista seuraavasti:

Metsälaki ei velvoita omistajaa hakkaamaan metsäänsä. Sen sijaan laki määrittää vähimmäisvaatimukset sille, miten metsää voi hakata ja hoitaa. Kasvatushakkuussa metsässä on säilytettävä riittävä, kehityskelpoinen puusto, uudistushakkuusta seuraa velvollisuus uuden metsän aikaansaamiseen. Lisäksi laki rajoittaa monimuotoisuuden kannalta arvokkaiden elinympäristöjen käsittelyä. Metsälaki koskee lähes kaikkia Suomen metsiä, lukuunottamatta metsälain 2 §:ssä mainittuja alueita.

Metsänomistajan on myös huolehdittava siitä, että kaikista hakkuista tehdään metsänkäyttöilmoitus Metsäkeskukselle vähintään 10 päivää ennen hakkuuta. Ilmoituksen laatii yleensä leimikon suunnittelija tai puunostaja, joka lähettää ilmoituksen Metsäkeskukseen.

Koska aihe sekä metsänkäyttöilmoitusten tekeminen on monelle Piimegan ohjelman käyttäjälle tärkeä osa työtä, päädyttiin kehitystyössä päätökseen uudesta ominaisuudesta, jolla voitaisiin korvata aikaisemmin käytössä ollut paperimuotoinen metsänkäyttöilmoitusten luontitapa. Ohjelmaan luotaisiin

ilmoitusten sähköinen ja mahdollisimman pitkälle automatisoitu luomismahdollisuus käyttäen hyväksi Metsäkeskuksen ylläpitämää sähköisten metsänkäyttöilmoitusten web-rajapintaa.

Metsänkäyttöilmoitusten luonti sähköisesti onnistui aiemminkin suoraan Metsäkeskuksen sivuilla, mutta se oli mahdollista vain syöttämällä kaikki tiedot sähköiselle lomakkeelle käsin. Opinnäytetyön projektissa otettaisiin käyttöön Metsäkeskuksen rajapinta ja esitäytettäisiin tiedot lomakkeelle mahdollisimman täydellisesti, mikä toisi selvän nopeushyödyn aiempaan prosessiin nähden riippumatta siitä, käytettiinkö aiemmin paperimuotoista vai web-sivulla täytettävää sähköistä lomaketta.

3 TYÖN TAVOITE JA VAATIMUKSET

Toteutettavan projektin arkkitehtuuri oli selvillä jo hyvin aikaisessa vaiheessa. Ensimmäisenä ja kaikkein tärkeimpänä vaatimuksena toteutukseen oli metsänkäyttöilmoitusten lähettäminen niin, että jokainen lähetys kulkisi käyttäjien tietokoneelta Piimegan keskuspalvelimen kautta Metsäkeskuksen rajapintaan. Keskuspalvelin haluttiin ehdottomasti saada mukaan sillä jokaisen ilmoituksen lähetysyritys oli järjestelmän seurannan kannalta erittäin tärkeää saada tallennettua. Tämänkaltainen keskitetty välityspalvelin toi etuna lähetysten seurantaan sen, että asiakkaasta riippumatta kaikki tieto lähetyksistä olisi aina samalla palvelimella eikä hajautettuna eri tietokantoihin. Myös ominaisuuden käyttöön ottaminen yksittäisille käyttäjille sekä lähettämiseen liittyvän koodin ylläpitäminen olisi huomattavasti helpompaa, sillä päivitettäviä ohjelmia olisi vain yksi monen sijasta.

Toisena vaatimuksena oli luoda Windowsilla toimivaan asiakasohjelmaan uusi ikkuna, josta metsänkäyttöilmoitusten lähettäminen tapahtuisi asiakkaan näkökulmasta. Uudella ikkunalla oli kaksi tärkeää käytettävyyteen liittyvää vaatimusta. Jokaisesta lähetyksestä piti tallentaa paikallisesti tieto lähetetystä sanomasta. Kun samaa hakkuualuetta tarkasteltiin myöhemmin kyseiseltä ikkunalta, oli käyttäjällä oltava nähtävissä kaikki kyseiseltä alueelta aikaisemmin lähetetyt sanomat. Jokaiseen aiemmin lähetettyyn sanomaan oli myös haluttaessa oltava mahdollista palata lataamalla kopio lähetetystä versiosta ja annettava sanoma jälleen käyttäjälle muokattavaksi sekä uudelleen lähetettäväksi.

Kolmantena päävaatimuksena oli kaikkien metsänkäyttöilmoituksen vaatimien tietojen täyttäminen etukäteen niin täydellisesti, kuin järjestelmässä olevien tietojen avulla oli suinkin mahdollista. Ideaalitapauksessa jokainen tieto olisi ilmoituksella jo valmiiksi oikein, kun ikkuna avattaisiin halutulta hakkuualuetta käsittelevältä näkymältä.

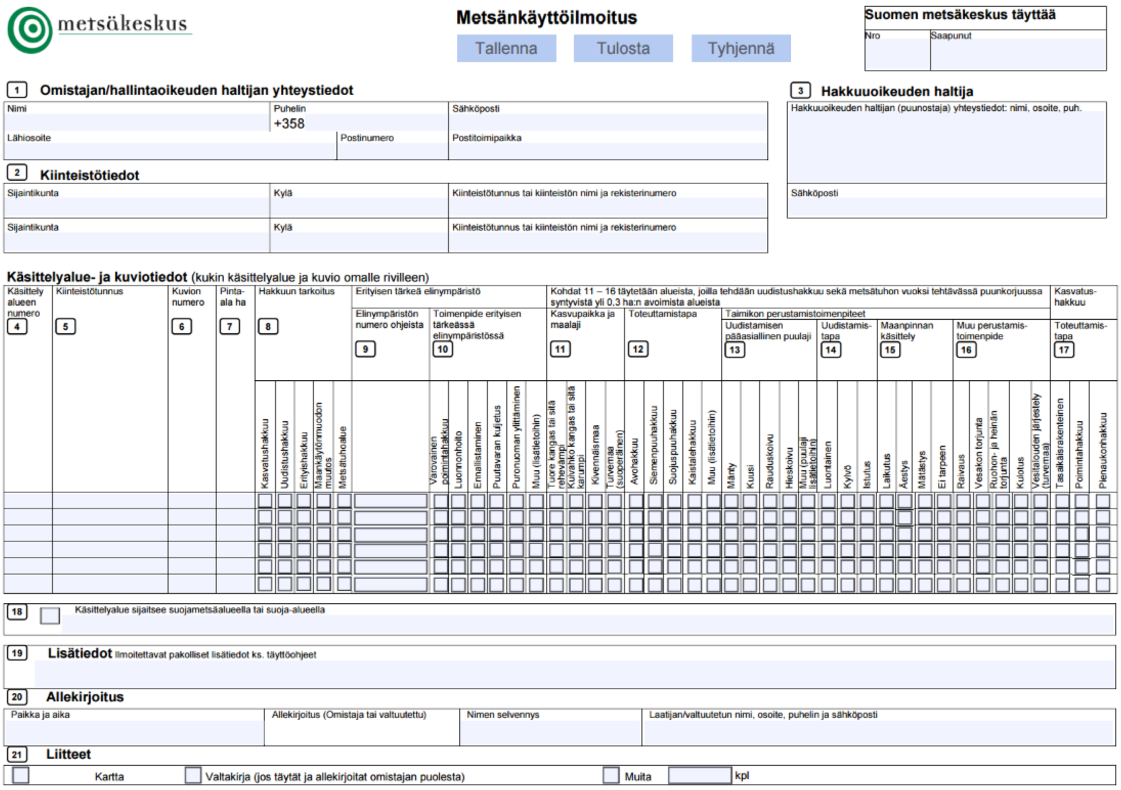
Tärkeänä vaatimuksena oli myös palvelun integroiminen Piimegan omaan laskutuspalveluun jonne kirjattaisiin jokainen viestin välitys. Kyseinen palvelu oli

jo olemassa valmiine rajapintoineen, joten tehtävänä oli luoda oma tunnus rajapintaa varten sekä varmistaa yhteyden toiminta sekä itse kutsun luonti rajapintaan.

4 OHJELMISTOKEHITYKSEN PROSESSIT JA TYÖTÄ KOSKEVAT KÄSITTEET

4.1 Metsänkäyttöilmoitus

Suomen metsälaki velvoittaa metsänomistajia tekemään kaikista myyntihakkuista metsänkäyttöilmoituksen viimeistään 14 päivää ennen hakkuun aloittamista. Metsänkäyttöilmoituksella (kuva 1) kerrotaan muun muassa hakkuun tarkoitus (kasvatushakkuu, uudistushakkuu, erityishakkuu, ...), käsiteltävän alueen pinta-ala ja toteuttamistapa.



metsäkeskus

Metsänkäyttöilmoitus

Tallenna | Tulosta | Tyhjennä

Suomen metsäkeskus täyttää

Nro | Saapunut

1 Omistajan/hallintaoikeuden haltijan yhteystiedot

Nimi | Puhelin +358 | Sähköposti

Lähiosoite | Postinumero | Postitoimipaikka

2 Kiinteistötiedot

Sijaintikunta | Kylä | Kiinteistötunnus tai kiinteistön nimi ja rekisterinumero

Sijaintikunta | Kylä | Kiinteistötunnus tai kiinteistön nimi ja rekisterinumero

3 Hakkuuoikeuden haltija

Hakkuuoikeuden haltijan (puunostaja) yhteystiedot: nimi, osoite, puh.

Sähköposti

Käsittelyalue- ja kuviotiedot (kukin käsittelyalue ja kuvio omalle rivilleen)

Käsittely alueen numero (4) | Kiinteistötunnus (5) | Kuvion numero (6) | Pinta-ala ha (7) | Hakkuun tarkoitus (8): Kasvatushakkuu, Uudistushakkuu, Erityishakkuu, Maankäyttömuodon muutos, Metsätuhoalue

Erityisen tärkeä elinympäristö: Elinympäristön numero ohjeista (9); Toimenpide erityisen tärkeässä elinympäristössä (10): Varovainen poimintahakkuu, Luonnonhoito, Ennallistaminen, Puutavaran kuljetus, Puronuoman ylittäminen, Muu (lisätietoihin)

Kohdat 11 – 16 täytetään alueista, joilla tehdään uudistushakkuu sekä metsätuhon vuoksi tehtävässä puunkorjuussa syntyvistä yli 0,3 ha:n avoimista alueista

Kasvupaikka ja maalaji (11): Tuore kangas tai sitä rehevämpi, Kuivahko kangas tai sitä karumpi, Kivennäismaa, Turvemaa (suoperäinen)

Toteuttamistapa (12): Avohakkuu, Siemenpuuhakkuu, Suojuspuuhakkuu, Kaistalehakkuu, Muu (lisätietoihin)

Taimikon perustamistoimenpiteet: Uudistamisen pääasiallinen puulaji (13): Mänty, Kuusi, Rauduskoivu, Hieskoivu, Muu (puulaji lisätietoihin); Uudistamistapa (14): Luontainen, Kylvö, Istutus; Maanpinnan käsittely (15): Laikutus, Äestys, Mätästys, Ei tarpeen; Muu perustamistoimenpide (16): Raivaus, Vesakon torjunta, Ruohon- ja heinän torjunta, Kulotus, Vesitalouden järjestely (turvemaa)

Kasvatushakkuu: Toteuttamistapa (17): Tasaikäisrakenteinen, Poimintahakkuu, Pienaukkohakkuu

18 Käsittelyalue sijaitsee suojametsäalueella tai suoja-alueella

19 **Lisätiedot** Ilmoitettavat pakolliset lisätiedot ks. täyttöohjeet

20 **Allekirjoitus**

Paikka ja aika | Allekirjoitus (Omistaja tai valtuutettu) | Nimen selvennys | Laatijan/valtuutetun nimi, osoite, puhelin ja sähköposti

21 **Liitteet**

Kartta | Valtakirja (jos täytät ja allekirjoitat omistajan puolesta) | Muita kpl

KUVA 1. Metsänkäyttöilmoitus (1, linkit Metsänkäyttöilmoitus).

4.2 Ohjelmistokehityksen vaiheet

Jokaisen onnistuneen ohjelmiston kehitysprosessi koostuu tietyistä vaiheista ja tehtävistä, joihin kaikkien ohjelmien kehitys voidaan jakaa niiden koosta tai kompleksisuudesta riippumatta. Nämä eri tehtävät voidaan jakaa seuraavasti:

1. **Kommunikointi.** Ennen kuin kehitystyö voidaan aloittaa, on erittäin tärkeää käydä ohjelman loppukäyttäjän, tilaajan sekä muiden kehittäjien kanssa keskustelua ohjelman käyttötarkoituksesta sekä tavoitteista, joita ohjelmalla halutaan saavuttaa. Minkä ongelman ohjelma pyrkii ratkaisemaan?
2. **Suunnittelu.** Suunnittelulla pyritään kartoittamaan etukäteen kehityksen vaatimat resurssit, selvittämään työn määrää ja määrittelemään kehitettävän projektin valmistumisen kannalta tärkeät tehtävät. Suunnittelussa arvioidaan muun muassa kehitystyön riskejä sekä aikataulua.
3. **Mallinnus.** Mallinnuksessa pyritään saamaan parempi kuva tehtävästä kokonaisuutena sekä hahmottamaan lopullisia toimivia osia, mikä auttaa ymmärtämään valittua työn lähestymistapaa tarkemmin sekä tarvittaessa korjaamaan sitä. Ohjelman arkkitehtuurin suunnittelu on olennaisena osana vaihetta, jonka vuoksi mallinnusvaiheessa voidaan luoda prototyyppejä ohjelman lopullisista eri osista.
4. **Rakennus.** Ohjelman rakennusvaihe käsittää ohjelman koodin luomisen käsin tai automatisoidusti generoimalla sekä luodun koodin testaamisen.
5. **Käyttöönotto.** Käyttöönotossa valmiiksi tai vaadittavan pitkälle kehitetty ohjelma toimitetaan asiakkaalle, joka arvioi työn tulosta ja antaa palautetta.

4.3 XML

XML on tekstimuotoinen standardoitu viestin muodostustapa, joka välittää tiedon avain-arvo pareista muodostettuina elementteinä. XML-viestien muodostukselle on tarkoin määritetyt ohjeet niiden rakenteesta ja sallituista merkeistä. Jotta viestit tulkitaan oikeellisiksi ja virheettömästi muodostetuiksi, pitää jokaisen viestin täyttää nämä ehdot ja noudattaa oikeita kielen rakenteita, avain-arvo-pareja ja elementtien määrittelyjä. Valmiiksi määritellyt säännöt auttavat virheiden minimoimisessa, kun jokaisen viestin sisältö on joissain määrin ennalta-arvattavissa (kuva 2).

```
▼<ForestUseDeclaration>
   <fud:UpdatePreviousDeclaration>0</fud:UpdatePreviousDeclaration>
   <fud:DeclarationReference>S-E20156492330917134159959</fud:DeclarationReference>
  ▼<fud:DeclarationTextInformation>
     Manuaalisesti generoitu metsänkäyttöilmoitusesimerkki. Paikkatieto esitetty suositellussa
   </fud:DeclarationTextInformation>
   <fud:SpecialPermission>0</fud:SpecialPermission>
  ▼<fud:CuttingRightsOwner language="fi">
     <ci:BusinessId>1039050-8</ci:BusinessId>
     <ci:OrganizationName>Stora Enso Oyj</ci:OrganizationName>
     <ci:Address>PL 309</ci:Address>
     <ci:PostalCode>00101</ci:PostalCode>
     <ci:PostOffice>HELSINKI</ci:PostOffice>
     <ci:CountryCode>FI</ci:CountryCode>
     <ci:CountryText>SUOMI / FINLAND</ci:CountryText>
     <ci:EmailAddress>mkipalaute@storaenso.com</ci:EmailAddress>
   </fud:CuttingRightsOwner>
  ▼<fud:CuttingsRightsOwnerRepresentative language="fi">
     <ci:FirstName>Olli</ci:FirstName>
     <ci:LastName>Ostaja</ci:LastName>
     <ci:PersonOrganizationName>Stora Enso Metsä</ci:PersonOrganizationName>
     <ci:Address>Savonkatu 47</ci:Address>
     <ci:PostalCode>79100</ci:PostalCode>
     <ci:PostOffice>LEPPÄVIRTA</ci:PostOffice>
     <ci:CountryCode>FI</ci:CountryCode>
     <ci:CountryText>SUOMI / FINLAND</ci:CountryText>
     <ci:MobilePhoneNumber>02046 47721</ci:MobilePhoneNumber>
     <ci:EmailAddress>olli.ostaja@storaenso.com</ci:EmailAddress>
   </fud:CuttingsRightsOwnerRepresentative>
  ▼<fud:Sender language="fi">
     <ci:FirstName>Olli</ci:FirstName>
     <ci:LastName>Ostaja</ci:LastName>
     <ci:PersonOrganizationName>Stora Enso Metsä</ci:PersonOrganizationName>
     <ci:Address>Savonkatu 47</ci:Address>
     <ci:PostalCode>79100</ci:PostalCode>
     <ci:PostOffice>LEPPÄVIRTA</ci:PostOffice>
     <ci:CountryCode>FI</ci:CountryCode>
     <ci:CountryText>SUOMI / FINLAND</ci:CountryText>
     <ci:MobilePhoneNumber>02046 47721</ci:MobilePhoneNumber>
     <ci:EmailAddress>olli.ostaja@storaenso.com</ci:EmailAddress>
     <fud:PowerOfAttorney>1</fud:PowerOfAttorney>
   </fud:Sender>
  ▼<fud:DeclarationRealEstates>
    ▼<re:RealEstateOwners>
      ▼<re:RealEstateOwner language="fi">
         <ci:FirstName>Matti</ci:FirstName>
         <ci:LastName>Metsänomistaja</ci:LastName>
         <ci:Address>Metsäkatu 2</ci:Address>
         <ci:PostalCode>70100</ci:PostalCode>
         <ci:PostOffice>KUOPIO</ci:PostOffice>
         <ci:CountryCode>FI</ci:CountryCode>
         <ci:CountryText>SUOMI / FINLAND</ci:CountryText>
         <ci:MobilePhoneNumber>0503376000</ci:MobilePhoneNumber>
         <ci:EmailAddress>matti.metsanomistaja@omasahkoposti.fi</ci:EmailAddress>
       </re:RealEstateOwner>
     </re:RealEstateOwners>
```

KUVA 2. Esimerkki tiedonsiirtopalveluun lähetettävästä XML-tiedostosta.

4.4 XSD

XML Schema Definition on tapa määritellä elementtejä ja tarkastella niiden rakennetta ja oikeellisuutta kokoelmana erilaisia tietueita (2, linkit Introduction -> Purpose).

Käytännössä katsoen, XSD-formaattia (kuva 3) käytetään tietorakenteiden määrittelyyn ja dokumentointiin. Yhteisellä määrittelystandardilla voidaan varmistaa, että kaksi eri osapuolta saavat samanlaisen tietorakenteen käyttöön, esimerkiksi kuvattaessa henkilön osoitetta, joka voi koostua useasta eri tietueesta kuten tie, postinumero ja maa.

```
<xs:complexType name="DeclarationStandType">
  <xs:annotation>
    <xs:documentation xml:lang="fi">Kuvion tiedot</xs:documentation>
    <xs:documentation xml:lang="sv"/>
    <xs:documentation xml:lang="en">Stand data</xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <xs:element ref="StandNumber" minOccurs="1" maxOccurs="1"/>
    <xs:element ref="StandNumberExtension" minOccurs="0" maxOccurs="1"/>
    <xs:element ref="DeclarationStandReference" minOccurs="1" maxOccurs="1"/>
    <xs:element ref="Area" minOccurs="1" maxOccurs="1"/>
    <xs:element ref="CuttingPurpose" minOccurs="1" maxOccurs="1"/>
    <xs:element ref="CuttingRealizationPractice" minOccurs="1" maxOccurs="1"/>
    <xs:element ref="ForestDamageQualifier" minOccurs="0" maxOccurs="1"/>
    <xs:element ref="HabitatCode" minOccurs="1" maxOccurs="1"/>
    <xs:element ref="HabitatOperations" minOccurs="0" maxOccurs="1"/>
    <xs:element ref="OtherHabitatCode" minOccurs="0" maxOccurs="1"/>
    <xs:element ref="MainGroup" minOccurs="0" maxOccurs="1"/>
    <xs:element ref="SubGroup" minOccurs="0" maxOccurs="1"/>
    <xs:element ref="FertilityClass" minOccurs="0" maxOccurs="1"/>
    <xs:element ref="SoilType" minOccurs="0" maxOccurs="1"/>
    <xs:element ref="DeclarationMainTreeSpecies" minOccurs="0" maxOccurs="1"/>
    <xs:element ref="DeclarationDevelopmentClass" minOccurs="0" maxOccurs="1"/>
    <xs:element ref="MeanAge" minOccurs="0" maxOccurs="1"/>
    <xs:element ref="MeanDiameter" minOccurs="0" maxOccurs="1"/>
    <xs:element ref="DeclarationRegenerationCommitment" minOccurs="0" maxOccurs="1"/>
    <xs:element ref="DeclarationRegenerationOperation" minOccurs="0" maxOccurs="1"/>
    <xs:element ref="DeclarationSoilPreparationOperation" minOccurs="0" maxOccurs="1"/>
    <xs:element ref="DeclarationOtherOperations" minOccurs="0" maxOccurs="1"/>
    <xs:element ref="DeclarationStandTextInformation" minOccurs="0" maxOccurs="1"/>
    <xs:element ref="gml:polygonProperty" minOccurs="1" maxOccurs="1">
      <xs:annotation>
        <xs:documentation xml:lang="fi">Metsänkäyttöilmoituksen kuvion aluegeometria sisäalueineen.</xs:documentation>
        <xs:documentation xml:lang="sv"/>
        <xs:documentation xml:lang="en">Polygon geometry of stand of forest use declaration with interior polygons.</xs:documentation>
      </xs:annotation>
    </xs:element>
    <xs:element ref="LocationEstates" minOccurs="1" maxOccurs="1"/>
  </xs:sequence>
  <xs:attribute name="id" type="co:IdStringType">
    <xs:annotation>
      <xs:documentation xml:lang="fi">Kuvion id</xs:documentation>
      <xs:documentation xml:lang="sv"/>
      <xs:documentation xml:lang="en"/>
    </xs:annotation>
  </xs:attribute>
</xs:complexType>
```

KUVA 3. Osa metsänkäyttöilmoituksen XML-tiedoston määrittelevää XSD-tiedostoa.

4.5 HTTP-metodit

HTTP-protokolla määrittelee joukon metodeja (tai verbejä), joiden avulla käyttäjä voi suorittaa erilaisia toimenpiteitä HTTP-protokollaa noudattavalla palvelimella. Yleensä toiminnot kohdistuvat esimerkiksi palvelimella sijaitseviin tiedostoihin, joita voidaan hakea GET-pyynnön avulla. Metodit voidaan jakaa kahteen eri luokkaan: turvallisiin pyyntöihin, joiden avulla haetaan tietoa eikä niistä seuraa sivuvaikutuksia palvelimelle sekä pyyntöihin (esimerkiksi POST), jotka voivat aiheuttavat muutoksia palvelimella.

HTTP-protokolla määrittelee muun muassa seuraavat metodit tai verbit:

GET: GET-metodilla voidaan pyytää tiettyä resurssia palvelimelta. Tämän kaltainen resurssi voi olla esimerkiksi kuva, tekstitiedosto tai HTML-sivu.

HEAD: HEAD-pyyntö vastaa tavallista GET-pyyntöä, mutta ilman tavallisesti GET-pyynnössä mukana tulevaa viestin body-osuutta. Viesti sisältää pelkästään otsikkotason metatietoa.

POST: POST pyytää palvelinta vastaanottamaan käyttäjän lähettämää tietoa yleensä ennalta määritellyssä formaatissa. Tämän kaltaisia formaatteja voi olla esimerkiksi joukko parametreja tai tietyn tyyppinen tiedosto.

DELETE: DELETE pyytää palvelinta poistamaan määritellyn resurssin.

OPTIONS: OPTIONS pyytää palvelinta listaamaan palvelimen tukemat HTTP-metodit määritellylle osoitteelle.

4.6 HTTP-autentikointi

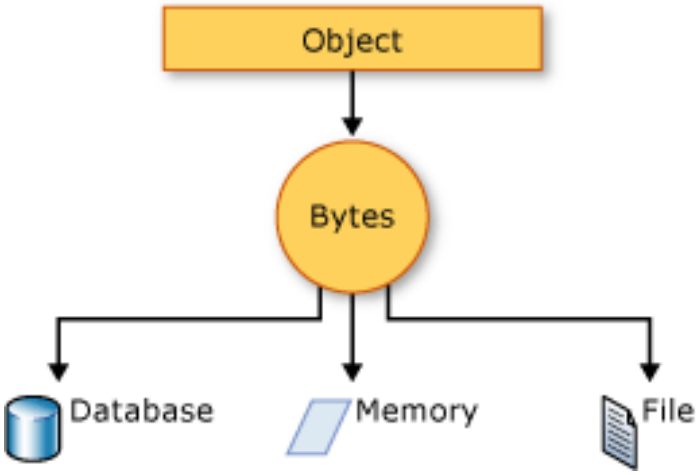
HTTP tarjoaa erilaisia vaihtoehtoja käyttäjän tunnistukselle palveluun. Yksi vaihtoehto on Basic Authentication (BA), jonka avulla palveluun vaadittava tunnus ja salasana voidaan lähettää sille user agentin erikseen omistetussa kentässä (3, s. 5), esimerkiksi:

```
Authorization: Basic QWxhZGRpbjpPcGVuU2VzYW1l
```

Tunnus ja salasana voidaan lähettää myös osana URL-osoitteen merkkijonoa. Molemmissa tapauksissa käytetään yleensä salattua yhteyttä, koska tunnistetietojen lähetys osana HTTP-pyyntöä ei itsessään suojaa niitä.

4.7 Serialisointi ja deserialisointi

Serialisoinnin ja sen vastaparin deserialisoinnin tarkoituksena on muuntaa tiedon formaattia. Serialisoinnissa tieto muunnetaan universaalimpaan ja geneerisempään formaattiin helpompaa tiedonsiirtoa varten erilaisten rajapintojen yli toiseen ympäristöön. Deserialisoinnissa prosessi tehdään toisin päin jolloin tieto saadaan palautettua ohjelmakohtaiseen muotoon. Serialisointia vaativia tiedonsiirtoja voi olla esimerkiksi olion tietojen siirto levylle tekstimuotoon tai HTTP-yhteyden yli toiselle tietokoneelle (kuva 5). Tämän kaltaisessa tapauksessa tieto muunnetaan hyvin ohjelmakohtaisesta tietorakenteesta kaikkien muiden ohjelmien helposti ymmärtämään tietoformaattiin.



KUVA 5. Objekti serialisoidaan tavuiksi, jotka kuvaavat itse tiedon lisäksi myös alkuperäisen tietotyypin formaattia ja versiota. Tämän jälkeen objekti voidaan tallentaa monen rajapinnan yli erilaisiin ympäristöihin (4, linkit How Serialization Works).

Deserialisointi voidaan ymmärtää serialisoinnin käänteisenä vastaparina, milloin geneerinen tavujono tai teksti voidaan lukea takaisin tiettyyn sovelluskohtaiseen muotoon (kuva 6).

```
Public Class Book
    Public Title As String
End Class

Public Sub ReadXML()
    Dim reader As New System.Xml.Serialization.XmlSerializer(GetType(Book))
    Dim file As New System.IO.StreamReader(
        "c:\temp\SerializationOverview.xml")
    Dim overview As Book
    overview = CType(reader.Deserialize(file), Book)
    Console.WriteLine(overview.Title)
End Sub
```

KUVA 6. Kiintolevyllä olevan XML-tiedoston deserialisointi takaisin ohjelmakohtaiseen Book-tietotyyppiin. (5, linkit Example).

4.8 Yksikkötestaus

Ohjelmistoalalla on kehitetty monenlaisia eri testausmalleja ohjelman toiminnan validointia varten. Tässä työssä käytössä tärkeimpänä testausmetodina oli niin kutsuttu yksikkötestaaminen. Tämänkaltaisen testaamisen tarkoituksena on ottaa ohjelman pienimpiä mahdollisia loogisia osia ja verrata osan toiminnan lopputulos määritellyillä ehdoilla odotettuun lopputulokseen. Testattavan osan toiminto voi olla esimerkiksi laskutoimitus kahdelle eri arvolle tai uusi tietokoneelle luotu tiedosto. Testausmenetelmän vaatimuksena on, että jokaiselle lähtötilanteelle voidaan asettaa jonkinlainen ennalta odotettava lopputulos jonka testattavan koodin pitäisi saavuttaa.

Käytännössä yksikkötestaus tapahtuu useimmiten kokoelmalla pienikokoisia testejä, jotka kutsuvat ohjelman käyttämiä funktioita (kuva 7). Koodin testikokoelmaa pyritään suorittamaan säännöllisin väliajoin, että ohjelman uusien virheiden ilmestyminen havaitaan tarpeeksi ajoissa. Suoritus voi tapahtua automaattisesti ohjelman jokaisella käynnistyskerralla, jokaisen koodin muutoksen jälkeen ohjelman versionhallinnassa tai aina jokaisen uuden ohjelmaversion julkaisun yhteydessä.

```
[TestMethod]
public void Withdraw_ValidAmount_ChangesBalance()
{
    // arrange
    double currentBalance = 10.0;
    double withdrawal = 1.0;
    double expected = 9.0;
    var account = new CheckingAccount("JohnDoe", currentBalance);
    // act
    account.Withdraw(withdrawal);
    double actual = account.Balance;
    // assert
    Assert.AreEqual(expected, actual);
}

[TestMethod]
[ExpectedException(typeof(ArgumentException))]
public void Withdraw_AmountMoreThanBalance_Throws()
{
    // arrange
    var account = new CheckingAccount("John Doe", 10.0);
    // act
    account.Withdraw(20.0);
    // assert is handled by the ExpectedException
}
```

KUVA 7. Esimerkki testistä, jossa varmistetaan kuvitteellisen pankkitilin oikea saldo noston jälkeen.

5 TYÖN TOTEUTUS

5.1 Kehitysprosessi

Projektin kehityksessä toteutuivat tyypillisen ketterän ohjelmistokehityksen vaiheet, mutta erityisesti suunnittelun merkitys korostui, sillä projektin toteutus vaati useiden eri toimintojen ja järjestelmien integrointia verkkoyhteyden yli toimiakseen.

Työn kehitys tapahtui käyttämällä niin kutsuttuja ketteriä kehitystapoja, joita noudatetaan Piimega Oy:llä kehitystyöskentelyssä. Prosessiin kuului Piimegalla kerran viikossa järjestettävä tilannepalaveri sekä tarkemmin työn edistystä seuraavat palaverit, joita pidettiin aina katsottaessa tilanteen sitä vaativan tai kun muuten tilannetta haluttiin kartoittaa. Tarkemmin käydyissä palavereissa sovittiin eri osien toteutuksesta teknisellä tasolla, ratkaistiin mahdollisia työssä ilmenneitä ongelmia sekä testattiin ohjelmaa loppukäyttäjän näkökulmasta, mihin kuului muun muassa ominaisuuden käytettävyyden testaus. Sähköisten metsänkäyttöilmoitusten perimmäisenä tarkoituksena oli saada ilmoitusten luonti mahdollisimman vaivattomaksi ja ennen kaikkea nopeaksi verrattuna perinteisiin paperille täytettäviin ilmoituksiin. Tämän vuoksi käytettävyystestaus oli erittäin tärkeää.

5.2 HTTP:n käyttö Metsäkeskuksen tiedonsiirtopalvelussa

Metsäkeskuksen tarjoaman tiedonsiirtopalvelun käyttämiseen vaadittiin erikseen rekisteröitävä tunnus ja salasana. Yhteys palveluun toimi salatulla HTTPS-yhteydellä eli salattua HTTP-yhteyttä käyttäen. Rekisteröitymisen jälkeen tunnistus palveluun toimi osana jokaista sanomaa, jossa muiden HTTP-parametrien mukana annettiin aiemmin luodut tunnistetiedot.

Metsänkäyttöilmoitusten siirtoon vaaditaan niin sanotulta clientilta eli käyttäjältä vain POST-metodia. POST-metodi sisältää parametreissa kaikki vaadittavat tiedot (taulukko 1), minkä vuoksi yhden metsänkäyttöilmoituksen lähetys onnistui yhdellä ainoalla HTTP-pyynnöllä. Tiedonsiirtopalvelu mahdollistaa jopa useamman kuin yhden metsänkäyttöilmoituksen lähetyksen samassa viestissä.

Kaikista lähetetyistä viesteistä palvelin lähettää synkronisesti takaisin palautesanoman, jossa ilmoitetaan, onnistuiko lähetetyn metsänkäyttöilmoituksen käsittely sekä mitä olivat mahdolliset siirrossa tapahtuneet virheet. Erilaiset virheviestit ilmoittavat syyn ongelmaan lähetyksessä, kuten esimerkiksi virheelliset tunnistetiedot tai vääränlaisessa formaatissa lähetetty XML.

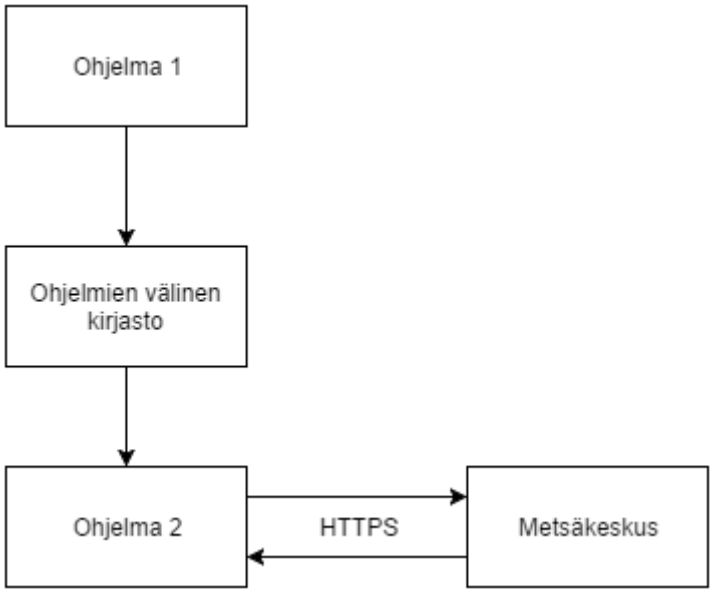
Parametri	**Parametrin arvo**
tunnus	Tiedonsiirtopalvelun käyttäjän käyttäjätunnus.
salasana	Tiedonsiirtopalvelun käyttäjän salasana.
td_nimi*n*	Siirrettävän tiedoston nimi. Parametrissa *n* = lähetyskohtainen siirrettävän tiedoston juokseva numero.
td*n*	Siirrettävän tiedoston (xml-dokumentin) sisältö. Parametrissa *n* = lähetyskohtainen siirrettävän tiedoston juokseva numero.
loppu	Lopetusmerkki. Merkkijonon loppuun parametri loppu, joka saa arvokseen ok.

TAULUKKO 1. Vaadittavat HTTP-parametrit metsänkäyttöilmoituksen lähetykseen.

Vastausviestit Metsäkeskuksesta tulivat synkronisena vastauksena POST-pyyntöön. Viestin sisältö oli XML-muodossa, josta se oli suoraviivaista deserialisoida helposti käsiteltäväksi olioksi lähetysviestien tavoin.

5.3 Tiedon välitys

Useiden eri tuotteiden vuoksi päätettiin tietoa välittävän ominaisuuden koodi keskittää ohjelmien väliseen komponenttiin, jota useampi ohjelma voisi kutsua (kuva 4). Tällä mahdollistettiin metsänkäyttöilmoituksiin liittyvän rajapinnan käyttämisen useammasta ohjelmasta.



KUVA 4. Tiedon kulku järjestelmässä.

5.4 Metsänkäyttöilmoituksen muokkaaminen

Vaikka valtaosin tiedot voitiin hakea järjestelmällä automaattisesti, käyttäjän valitsemalle puukauppasopimukselle oli ohjelman käyttäjällä kuitenkin oltava mahdollisuus metsänkäyttöilmoituksen yksittäisten kenttien muokkaukseen, lähetettävien tietojen kokonaisuuksien esikatseluun sekä lisätietojen antamiseen Metsäkeskukselle lähetettävään metsänkäyttöilmoitukseen. Ohjelman yleisen joustavuuden lisäksi metsänkäyttöilmoitusten muokkaamismahdollisuuden toisena perusteluna oli järjestelmän tietokannan rajoite puukauppasopimusten tietojen suhteen: kaikkia tarvittavia tietojen erittelyjä ei ollut toteutettu tietokantaan, mikä esti tiedonhaun täydellisen automatisoinnin. Vaatimus päätettiin ratkaista luomalla järjestelmään oma hallintaikkuna metsänkäyttöilmoituksen lähetyksen yhteyteen.

5.5 Lähetettyjen tietojen lokitus

Koska lähetettävät tiedot siirtyivät Piimegan ulkopuoliseen palveluun, lähetetyistä tiedoista oli pidettävä kirjaa, että vältyttäisiin aiempien tietojen turhalta uudelleenlähettämiseltä sekä vahingolliselta aiempien tietojen korvaukselta. Lähetettävien tietojen kirjaus hoidettiin tallentamalla jokainen lähetetty sanoma aikaleiman kanssa Microsoft SQL Server -tietokantaan.

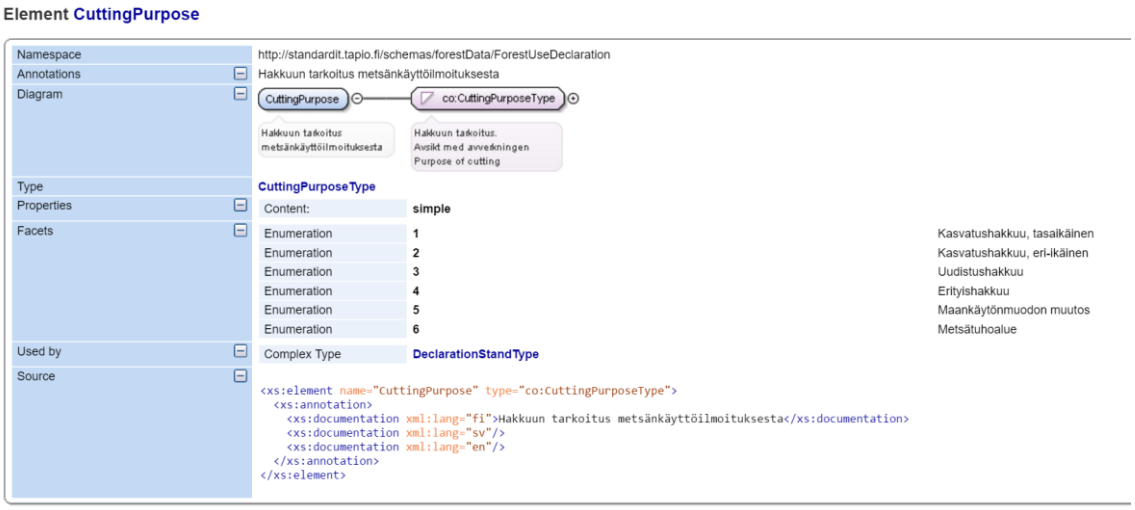
Microsoft SQL Serverillä on natiivi tuki XML-tietotyypille, mikä mahdollisti jokaisen dokumentin helpon arkistoinnin tietokantaan.

5.6 Serialisointi ja deserialisointi

Metsänkäyttöilmoituksia lähetettäessä ohjelmakohtainen tietotyyppi serialisoitiin puhtaaseen tekstimuotoon HTTP-pyyntöä tehdessä. Microsoftin .NET-ohjelmakirjastoa hyväksi käyttäen saatiin HTTP-pyynnön niin sanottuun body-osuuteen laitettua tekstimuotoinen versio koko metsänkäyttöilmoituksesta. Samoin HTTP-pyynnön vastaussanomasta saatiin tekstimuotoinen paluuarvo käännettyä objektiksi, josta ohjelman oli helppo käsitellä vastauksen eri kenttiä ja arvoja.

5.7 XML-rakenteen integrointi

Tärkeää oli tietää, mitä tietoja XML-tiedostoon kuului hakea ja minkälainen rakenne sillä oli, että Metsäkeskuksen rajapinta hyväksyisi lähetetyt sanomat. Tämän vuoksi oli tarpeellista tutkia sanoman rakennetta virallisesta dokumentaatiosta ja lukea jokaisen tietueen selvennys, joka määrittelee, mitä tietoa kyseinen kenttä käytännössä sisälsi. XSD-tiedosto ei itsessään määrittele tietueen mahdollisia laillisia arvoja, vaan kertoo ainoastaan tietueen tyypin, joka voi olla merkkijono, lukuarvo tai muu yleisesti ohjelmointikielissä käytetty datatyyppi. Metsäkeskuksen sanomat ovat Bitcomp Oy:n määrittelemiä. Virallinen dokumentaatio työssä käytettävälle sanomalle löytyi osoitteesta http://bitcomp.fi/metsastandardi_ehdotus/V8/MKS/doc/index.html (6, linkit ForestUseDeclaration).



KUVA 8. Dokumentaatio kertoo CuttingPurpose-kentän sisältämän tyypin, tiedon sekä sen saamat mahdolliset arvot.

Viestin rakenteen selvittäminen etukäteen oli tärkeää suorittaa huolellisesti, sillä kuten monissa integraatioprojekteissa, tietojen toimiva yhteensovittaminen eri järjestelmistä on tärkeää järjestelmän myöhemmän ylläpidon kannalta. Näin huolehditaan, että jatkokehitys tapahtuisi mahdollisimman vaivattomasti ja että ongelmia ei kohdattaisi myöhemmin väärien valintojen vuoksi. Projektin aikana järjestettiin viikoittain vanhemman kehittäjän kanssa useita palavereja, joissa kartoitettiin integraation etenemistä sekä selvitettiin mahdollisia ongelmakohtia tietojen täyttämisessä Metsäkeskuksen järjestelmään.

Kun viestin rakenne ja sen vaatimat tiedot oli kartoitettu Bitcompin dokumentaation avulla, oli vuorossa XML-rakenteen integroiminen ohjelman koodiin. XML-rakenteessa käytettiin tavanomaisten tietotyyppien lisäksi erittäin paljon Metsäkeskuksen itse määrittelemiä XML-elementtejä. Tästä johtuen XML-rakenne kokonaisuudessaan oli erittäin suuri ja sen määrittelevä XSD-tieto olikin pilkottu moneen eri tiedostoon. Näin suuren XSD-luokkakokonaisuuden integroiminen käsin kopioimalla olisi ollut liian aikaa vievä työ, joten integroimisessa käytettiin hyväksi Microsoftin kehittämää XSD-työkalua. Komentorivityökalulla onnistui XSD-tietojen pohjalta tapahtuva VB.Net–luokkien generointi, jotka toteuttivat XSD-tietojen esittelemät rakenteet. Kun nämä VB.Net-luokat deserialisoitiin ohjelmassa, saatiin aikaiseksi Metsäkeskuksen

haluamia XML-tiedostoja jotka vastasivat myös Metsäkeskuksen antamaa esimerkkitiedostoa lähetettävästä metsänkäyttöilmoituksesta (kuva 2).

5.8 Testaus

Ohjelmointitasolla testaamiseen kuului tärkeänä osana Metsäkeskuksen testikäyttöön tarkoitettu rajapinta, joka toiminnaltaan vastasi täysin oikeaa tuotantokäyttöön tarkoitettua rajapintaa. Testirajapinnasta vastauksena jokaiseen lähetykseen saatiin validoinnin tuloksena kaikki mahdolliset virheet jotka lähetetty viesti sisälsi, tarkkojen virhekoodien sekä kuvausten kanssa. Projektissa ensimmäisenä osana toimintakuntoon saatettiin itse XML-viestin lähettävä rajapinta, jolloin koska vain lähes koko kehityskaaren ajan pystyttiin testaamaan sillä hetkellä olemassa olevan järjestelmän toimivuutta lähettämällä Metsäkeskuksen oma esimerkkiviesti testirajapintaan. Näin voitiin vahvistaa järjestelmän toimivuus ja viestin kulkeminen koko lähetysketjun läpi.

Omana toteutuksena Metsäkeskuksen suorittaman virheentarkistuksen lisäksi testaamisessa käytettiin hyväksi aiemmassa kappaleessa kuvattua yksikkötestausta. Ohjelmakoodiin rakennettiin useampi virheentarkistusta suorittava metodi, jotka jokaisen viestin oli läpäistävä ennen kuin lähettäminen tapahtuisi. Näillä testeillä varmistuttiin siitä, että jokainen viesti sisälsi kaikki tarvittavat tiedot ja että ne olisivat oikeassa muodossa. Virheen löytyessä lähetys lopetettiin ja siitä annettiin ilmoitus käyttäjälle. Kehitysvaiheessa myös vähemmän tärkeitä lähetyksen aikana suoritettavia toimenpiteitä testattiin samoilla menetelmillä.

6 YHTEENVETO

Opinnäytetyön tavoitteena oli luoda automaattinen metsänkäyttöilmoitusten luonti- ja lähetysjärjestelmä olemassa olevaan toiminnanohjausjärjestelmään.

Tarkoituksena oli luoda ensin ohjelman käyttäjälle avautuva ikkuna, josta käyttäjä pystyy luomaan yhden napin painalluksella esitäytetyn metsänkäyttöilmoituksen, jota haluttaessa voitaisiin muokata. Työn toinen osa oli rajapinnan luonti, joka automaattisesti lähettäisi aiemmassa vaiheessa luodun metsänkäyttöilmoituksen Metsäkeskuksen vastaavaan rajapintaan sekä tallentaisi lähetettyjen metsänkäyttöilmoitusten tiedot myöhempää tarkastelua varten. Käyttäjällä piti olla myös mahdollisuus palata mihin tahansa aiemmin lähetettyyn versioon lähetyshistoriaa tarkastelemalla.

Työssä suuressa roolissa olivat HTTP-protokolla ja sen metodit, koska toiminnallisuuden ydin oli XML-tiedostojen lähetys HTTP-rajapintaan sekä serialisointi, jota sovellettiin tietojen muuttamisessa tietojen lähetys- ja vastaanottovaiheessa sopivaan muotoon.

Vaikka työn alussa itselläni olikin jonkin verran kokemusta käytettävistä menetelmistä, kuten HTTP-kutsuista ja XML-serialisoinnista, opetti projekti silti huomattavasti muun muassa serialisoinnista sekä XML-tiedostojen syntaksista. Metsäkeskuksen rajapinnassa tapahtunut lähetettyjen tiedostojen validointi tarkoitti, että tiedostojen generoinnin ja niiden sisällön piti vastata täsmälleen ennalta määritettyä muotoa ja sisältää kaikki vaaditut XML-attribuutit sekä kentät. Alussa lähdin tekemään XML-tiedostojen vastaavia luokkia koodiin lukemalla dokumentaatiota ja kirjoittamalla koodia käsin, mutta tämä osoittautui nopeasti epäkäytännölliseksi ja virhealttiiksi menetelmäksi, koska projektin ja rajapinnan käyttämiä XML-tiedostoja piti määritellä ohjelman koodiin kymmenittäin. Tehtävän mahdollistamiseksi löytyi ohjelmatyökalu nimeltä XSD, jonka avulla työ helpottui huomattavasti, koska rajapinnan määrittelysivuilta löytyneen esimerkkisanoman perusteella työkalulla saatiin generoitua kaikkien vaadittavien luokkien pohja. Generointi ei täydellisesti luonut kaikkia luokkia, ja siinä piti ymmärtää, miten koodissa muutokset luokkiin vaikuttivat

lopputulokseen, kun luokka serialisoitiin XML-tiedostoksi asiakkaan oikeilla tiedoilla. Tässä kohtaa XML-tiedostojen dokumentaatioon perehtyminen auttoi ymmärtämään vaadittavia muutoksia ohjelmakoodiin.

HTTP-kutsujen käyttäminen ja lähetettävien kutsujen tutkiminen eri työkaluilla esimerkiksi virhetilanteissa auttoi ymmärtämään viestien käytännön toiminnan sekä sisällön, joka oli aiemmin ollut itselleni hankalaa.

Kokonaisuudessaan projektin teossa suurena apuna oli aiempi kokemukseni lähes kaikista käytetyistä teknologioista. Tämä auttoi nopeasti hahmottamaan ohjelmaan vaadittavan koodin sopivan toteutuksen ja rakenteen, jolloin jäljelle jäi yksityiskohtiin syventyminen, kuten mainittu XML-luokkien generointi.

Ominaisuus saatiin kehitettyä toimintakuntoon, mutta käyttöönotto asiakkaalle jäi myöhemmälle ajankohdalle. Ominaisuutta kehitetään mahdollisesti lisää myöhemmin.

LÄHTEET

1. Metsäkeskus. Lomakkeet. Saatavissa: https://www.metsakeskus.fi/lomakkeet. Hakupäivä 27.11.2017.

2. W3C XML Schema Definition Language. W3C Recommendation 5 April 2012. 2012. World Wide Web Consortium. Saatavissa: http://www.w3.org/TR/xmlschema11-1/. Hakupäivä 23.04.2016.

3. Franks, J. et al. HTTP Authentication: Basic and Digest Access Authentication. The Internet Engineering Task Force. 1999. Saatavissa: https://tools.ietf.org/html/rfc2617. Hakupäivä 1.5.2016.
4. Microsoft Corporation. Serialization (Visual Basic). 2015. Saatavissa: https://msdn.microsoft.com/en-us/library/mt656712.aspx. Hakupäivä: 15.5.2016.
5. Microsoft Docs. How to: Read Object Data from an XML File. 2015. Microsoft Corporation. Saatavissa: https://msdn.microsoft.com/en-us/library/mt656710.aspx. Hakupäivä 15.5.2016.
6. Bitcomp Oy. Main schema ForestCentreMessage.xsd. 2016. Saatavissa: http://bitcomp.fi/metsastandardi_ehdotus/V8/MKS/doc/index.html. Hakupäivä 1.6.2016.